# *Metodología de Desarrollo*

*Predicción del Éxito de Atracciones Turísticas: Un Enfoque de Deep Learning.*

*Patrocinador: Luis Mata Aguilar, Chief Technology Officer de la plataforma Artgonuts.*
*Desarrollado por: Orionis Di Ciaccio*

*Junio 2025*

## *Modelo Multimodal para la Predicción del Engagement Turístico*

*Este proyecto tiene como objetivo diseñar e implementar un modelo de Deep Learning capaz de predecir el nivel de engagement (alto o bajo) que generarán distintos Puntos de Interés (POIs) turísticos.*

*El modelo propuesto integra de forma innovadora dos tipos de información complementaria: características visuales de imágenes asociadas a cada POI y metadatos estructurados que describen su contexto.*

*La metodología se estructura en 6 grandes fases:*

1. ***Preparación del dataset multimodal***
2. ***Diseño del modelo***
3. ***Proceso de entrenamiento y evaluación.***
4. ***Registro de proceso experimental***
5. ***Posibles implementaciones de mejoras***
6. ***Conclusiones***

---

## *1. Preparación del Dataset Multimodal*

### *1.1 Estructura de Datos:*

- ***Imágenes:*** *cada POI cuenta con una imagen representativa almacenada en una subcarpeta individual dentro de un directorio principal.*

- ***Metadatos (CSV):*** *contiene atributos clave como:*

★ ***id****: Identificador único para cada punto de interés.*
★ ***name****: Nombre descriptivo del POI.*
★ ***shortDescription****: Breve descripción textual del punto de interés.*
★ ***ubicación geográfica:*** *Coordenadas (locationLon, locationLat), barrio, distrito.*
★ ***categorización:*** *Columna "categories" que indica el tipo de atracción o temática.*
★ ***tier:*** *Clasificación por nivel de relevancia o popularidad.*

- ★ ***tags:*** *Etiquetas descriptivas asociadas al POI.*
- ★ ***xps:*** *Experiencia obtenida por el usuario al visitar el POI (para gamificación)*

***Métricas de engagement:*** *Indicadores cuantitativos de la interacción de los usuarios con cada POI:*

- ★ *Visits: Número de visitas registradas.*
- ★ *Likes y Dislikes: Cantidad de valoraciones positivas y negativas recibidas.*
- ★ *Bookmarks: Número de veces que el POI ha sido añadido a favoritos.*

### *1.2 Asociación Imagen-Metadatos*

*Se diseñó un proceso de vinculación que empareja cada imagen con su correspondiente fila en el archivo CSV mediante un identificador común, en este caso a través de la columna* ***['main_image_path']***

### *1.3 Análisis Exploratorio de datos (EDA)*

*Examen detallado de la distribución y características de los datos, identificación de patrones, correlaciones y posibles anomalías, distribuido en 3 etapas:*

***I. EDA de los datos:***

- *Dimensiones (número de columnas y filas).*
- *Tipo de los datos (object, float, int)*
- *Valores faltantes o nulos (listas vacías)*
- *Estadisticas descriptivas ( media, mediana y desviacion estandar)* ***II. EDA de las imágenes:***
- *Inspección del shape de las imágenes (128×128×3)*
- *Tipo de dato (NumPy array)*

***III. Visualización de características geo espaciales interactivas***

### *1.4 Preprocesamiento y Feature Engineering*

***Variable objetivo:***

*se definió una etiqueta binaria* ***['engagement_label']*** *a partir de un* ***[engagement_score]****, que integra múltiples métricas (likes, visitas, bookmarks) categorizado como alto o bajo. En caso de que los valores estuvieran en formato numérico, se procedió a binarizarlos mediante un umbral de decisión. Los metadatos fueron sometidos a procesos de normalización y codificación (e.g., One-Hot Encoding para variables categóricas).*

***Observaciones de variables:***

➔ ***'Tier'****: Las redes neuronales aprenden representaciones internas y generalmente no hay problema en dejar los valores enteros ordinales tal como están, especialmente si se hace normalización del resto de las variables numéricas.*

➔ ***'Tags'****:¿Está bien tener 65 columnas de tags? Las redes neuronales manejan bien la alta dimensionalidad, sin embargo, filtraremos a solo las mas frecuentes y comprobaremos su importancia en la validacion.*

➔ ***'locationLon' y 'locationLat':*** *¿Por qué incluir latitud y longitud? Estas variables representan ubicación espacial del POI y pueden aportar mucha información contextual al modelo como: Zonas más céntricas podrían tener mayor engagement y lugares cerca de otros POIs populares podrían ser más visitados; por lo que experimentaremos con cluestering.*

I. ***Tratamiento de valores faltantes***

II. ***Codificación de variables categórica:*** *extraccion de tags por frecuencia como feature binario*

III. ***Creación de métricas:*** ***['engagement_score']*** *: convirtiendolo en un problema de clasificacion binaria y clasificación multiclase:*

- *1 = engagement alto*
- *0 = engagement bajo*

---

- *0 = Muy bajo engagement*
- *1 = Bajo engagement*
- *2 = Alto engagement*
- *3 = Muy alto engagement*

IV. ***Tratamiento de imágenes:*** *Calculo de Media y Desviacion estandar (se creo un bucle, y fue muy costoso 10 minutos)*

- *valores de media y STD: Muestra que las imágenes no tienen mucho ruido de color y están bastante bien distribuidas:*

***Media:([0.4451, 0.4412, 0.4202])***
***STD:([0.1908, 0.1909, 0.2015])***

- *Si las imágenes estuvieran muy contrastadas, la desviacion estandar subiría (ej. > 0.25).*
- *Si las imágenes estuvieran en blanco y negro, los dos canales [mean, std] estarían muy parecidos, incluso idénticos.*
- *Conversión a tensor y normalización con media y desviación estándar del dataset para mejorar estabilidad numérica y rendimiento en entrenamiento* ***V. Reduccion de Columnas:***

***['categories', 'categories_list', 'main_image_path','name','id', 'shortDescription','tags_list', 'n_tags','tags','Visits', 'Likes', 'Dislikes', 'Bookmarks']***

💡 ***Nota:*** *se evitó el uso de columnas que formaban parte del cálculo del engagement, para evitar fuga de información.*

### *1.5 División del Conjunto de Datos*

*El dataset completo fue dividido en tres subconjuntos:*

- ***Entrenamiento (70%)***
- ***Validación (15%)***
- ***Prueba (15%)***

*Esta división se realizó de forma estratificada para mantener balance entre las clases.*

---

## *2. Diseño del Modelo Multimodal*

*El modelo propuesto se basa en una arquitectura **multimodal de fusión temprana**, que integra las salidas intermedias de dos subredes*

### *2.0 Arreglos previos*

- ***Normalización de Datos Geográficos:** Las coordenadas **'locationLat'** y **'locationLon'** están en escalas numéricas distintas al resto de los metadatos. La normalización asegura que estas variables no dominen el aprendizaje del modelo debido a su escala. Además, facilita la convergencia del optimizador y mejora el rendimiento del modelo al hacer que todas las variables numéricas estén centradas.*

- ***Data Augmentation:** Técnicas como **RandomHorizontalFlip, RandomRotation, ColorJitter** introducen pequeñas variaciones en las imágenes durante el entrenamiento, simulando nuevas muestras.Esto enriquece el aprendizaje de la red convolucional, mejora la generalización, y permite que el modelo sea más robusto ante variaciones reales del entorno.*

- ***Creacion de clase personalizada:** La clase **'POIDataset'** Permite crear un dataset compatible con un DataLoader, se encarga de cargar la imagen asociada a cada POI y de extraer los metadatos relevantes del DataFrame.*

- ***Creacion de DataSets y DataLoader:** Una vez creada la clase **POIDataset**, se utilizan **random_split** o particiones estratificadas para dividir los datos en **entrenamiento, validación y test**.*

  *Los **DataLoaders** de PyTorch se encargan de:*

  ➔ *Leer los datos por lotes **(batch_size)***
  ➔ *Mezclarlos **(shuffle=True)** en entrenamiento para evitar aprender el orden* ➔ *Permiten que el entrenamiento sea escalable, reproducible y eficiente.*

### *2.1 Red 2.1 Red Visual (CNN personalizada)*

*Se desarrolló una red convolucional profunda construida desde cero, encargada de procesar imágenes representativas de cada punto de interés (POI). Esta red extrae automáticamente características visuales relevantes —como composición, textura, color y estructuras— que pueden estar asociadas con el atractivo visual del lugar y su nivel de engagement. La arquitectura incluye capas de convolución, activación ReLU, y reducción espacial mediante MaxPooling, finalizando con una capa totalmente conectada que resume la representación visual.*

### *2.2 Red Tabular (Metadatos estructurados)*

*En paralelo, se diseñó una red neuronal densa (fully connected) para procesar metadatos numéricos relacionados con los POIs. Estos atributos incluyen información contextual como ubicación geográfica normalizada, nivel de relevancia turística, categorías y etiquetas, entre otros. Esta subred modela el impacto de las características no visuales en el engagement, como la accesibilidad, clasificación oficial o experiencia ofrecida.*

### *2.3 Integración Multimodal*

*Las representaciones aprendidas por ambas subredes (visual y tabular) se concatenan y se introducen en una red de clasificación conjunta. Esta capa de fusión permite que el modelo combine de forma eficiente las señales visuales y contextuales, explotando la complementariedad entre ambas modalidades para mejorar la capacidad predictiva del sistema.* ***Se entrenaron dos versiones del modelo:*** *una para* ***clasificación binaria*** *(engagement alto/bajo) y otra para* ***clasificación multiclase*** *(muy bajo, bajo, alto, muy alto).*

---

## *3. Entrenamiento y Evaluación del Modelo*

### *3.1 Entrenamiento por Épocas (train_epoch / eval_epoch)*

*El proceso de entrenamiento se estructuró en ciclos de épocas, donde el modelo itera completamente sobre el conjunto de entrenamiento. En cada época, se ejecuta la función train_epoch, que realiza una pasada por todos los lotes (batches), actualizando los pesos de la red mediante retropropagación y optimización con el algoritmo Adam. Durante esta fase se calculan métricas clave como la pérdida promedio y la precisión sobre los datos de entrenamiento.*

*Simultáneamente, la función eval_epoch evalúa el rendimiento del modelo sobre el conjunto de validación al final de cada época, sin actualizar los parámetros. Esta evaluación periódica permite detectar signos de sobreajuste o bajo aprendizaje de forma temprana y ajustar la estrategia de entrenamiento si es necesario.*

### *3.2 Entrenamiento Completo (train_model)*

*La función **train_model** orquesta todo el proceso de entrenamiento durante múltiples épocas. En cada iteración:*

- *Ejecuta train_epoch para optimizar el modelo sobre los datos de entrenamiento.*
- *Evalúa el desempeño en validación con eval_epoch.*
- *Registra métricas como loss y accuracy en ambas fases.*

*Este enfoque estructurado permite realizar un seguimiento detallado del aprendizaje del modelo a lo largo del tiempo, facilitando la comparación de versiones, visualización de curvas de entrenamiento, y aplicación de técnicas de early stopping o ajuste de hiperparámetros en futuras iteraciones.*

### *3.3 Evaluación Final del Modelo (test_model)*

*Una vez finalizado el entrenamiento, se ejecuta la función test_model para evaluar el rendimiento del modelo sobre el conjunto de prueba no visto durante el entrenamiento ni la validación. Esta evaluación proporciona una estimación realista del poder predictivo del sistema en un entorno de producción.*

*La evaluación incluye el cálculo de métricas de clasificación (como accuracy y F1-score) y la generación de un informe detallado (classification_report) con el desempeño por clase, útil especialmente en el escenario de clasificación multiclase (muy bajo, bajo, alto, muy alto).*

***EL flujo de trabajo se resumen en:***

1. *Dataset → POIDataset*
2. *Transformaciones → transforms.Compose*

3. *DataLoaders → train / val / test*
4. *Definir redes → VisualCNN + TabularNN + MultimodalNet*
5. *Entrenar por épocas → train_epoch / eval_epoch*
6. *Entrenamiento completo → train_model*
7. *Evaluación final → test_model*

### *3.4 selección de hiperparámetros*

*Se establecieron valores iniciales para parámetros clave y se experimentó con distintas* ***épocas*** *y* ***tasas de aprendizaje.***

I. ***Función de pérdida:*** ***nn.BCEWithLogitsLoss()*** *se utilizó la función Binary Cross Entropy con logits en el modelo de clasificación binaria. Esta función combina de manera eficiente una capa sigmoide con la entropía cruzada binaria en una sola operación, lo cual:*
   - *Mejora la estabilidad numérica.*
   - *Aumenta la eficiencia del entrenamiento.*
   - *Es adecuada para problemas de clasificación binaria donde la etiqueta es 0 o 1.*

II. ***Optimizador:*** ***torch.optim.Adam()*** *se utilizó Adaptive Moment Estimation, que es robusto, eficiente y funciona bien en la mayoría de los escenarios, especialmente en redes neuronales profundas con arquitectura combinada (visual y tabular).*

III. ***Tasa de aprendizaje:***
   - *(1e-4) pueden llevar a un aprendizaje excesivamente lento o quedar atrapado en mínimos locales*
   - *(0.001) suelen ofrecer un buen compromiso entre velocidad de convergencia y estabilidad.*
   - *(0.01) pueden provocar inestabilidad en el entrenamiento o impedir la convergencia (saltos erráticos en el error).*

IV. ***Número de épocas****:* ***num_epochs= 15, 20, 50***

V. ***Tamaño del batch size****:* ***batch_size= 32.*** *Este valor representa un equilibrio entre eficiencia computacional y estabilidad del gradiente:*
   - *Batches pequeños → ruido alto en la estimación del gradiente.*
   - *Batches grandes → necesitan mucha memoria y pueden converger peor.*

### *3.5 Interpretación de los valores*

*Cada línea representa una época de entrenamiento, mostrando:*

- ***Train Loss:*** *Error promedio del modelo sobre los datos de entrenamiento.*
- ***Train Acc:*** *Precisión (accuracy) del modelo sobre el conjunto de entrenamiento.*
- ***Val Loss****: Error sobre los datos de validación (nunca vistos durante el entrenamiento).* ● ***Val Acc:*** *Precisión sobre el conjunto de validación.*

***Indicadores generales de comportamiento:*** *Durante el proceso de entrenamiento, se evaluaron distintas métricas para monitorear el aprendizaje del modelo y su capacidad de generalización. Algunas señales interpretativas clave son:*

➔ *Si el* ***Train Loss*** *disminuye progresivamente, esto indica que el modelo está aprendiendo a minimizar el error cometido sobre los ejemplos del conjunto de entrenamiento. Es señal de que la función de pérdida está guiando eficazmente el proceso de optimización.*

➔ *Si la* ***Val Accuracy*** *mejora durante las primeras épocas, sugiere que el modelo es capaz de generalizar más allá de los datos con los que fue entrenado, es decir, que está captando patrones útiles que se reproducen en datos nuevos.*

➔ *Si la* ***Train Accuracy*** *y la* ***Val Accuracy*** *se mantienen próximas entre sí, usualmente indica que el modelo no está sobreajustando (overfitting). En cambio, si la precisión en entrenamiento sube mucho pero la de validación se estanca o cae, podría estar memorizando los datos en lugar de aprender de forma generalizable.*

➔ *Si la* ***Validation Loss*** *también disminuye, es una señal de aprendizaje saludable. Significa que el modelo está mejorando no solo en aciertos sino también en su confianza al clasificar correctamente.*

---

## *4. Registro del Proceso Experimental*

*A lo largo del desarrollo del proyecto, se llevó a cabo un registro exhaustivo y sistemátic**o** de todas las fases experimentales. Este proceso permitió asegurar la trazabilidad, justificar las decisiones técnicas y construir una base sólida para validar científicamente la solución final. Se documentaron los siguientes aspectos:*

***Diseño de experimentos:***
*Se registraron todas las configuraciones evaluadas, incluyendo:*

➢ *Variantes arquitectónicas (estructuras de redes visuales, tabulares y de fusión).*
➢ *Cambios en la selección de atributos tabulares.*
➢ *Tipos de funciones de activación, normalización y técnicas de regularización.*

- ***Hiperparámetros:***
  *Se evaluaron diferentes combinaciones de:*
  - *Tasa de aprendizaje.*
  - *Número de épocas.*
  - *Optimizadores (Adam,).*

- *Estrategias de partición de datos.*

- ***Resultados cuantitativos:***
  *Para cada experimento se recopilaron métricas clave:*
  - *Precisión (accuracy), pérdida (loss) y F1-score.*
  - *Curvas de aprendizaje (train vs. validation).*
  - *Reportes de clasificación.*

- ***Observaciones cualitativas:***
  *Se analizaron errores recurrentes y casos problemáticos, lo que permitió:*
  - *Identificar sesgos del modelo.*
  - *Detectar clases mal representadas.*
  - *Proponer mejoras en la representación multimodal.*

- ***Errores, fracasos y aprendizajes:***
  *Se documentaron fallos relevantes como:*
  - *Sobreajuste por arquitecturas muy profundas.*
  - *Bajo rendimiento al omitir la normalización de coordenadas.*
  - *Experimentos descartados por no mejorar la generalización.*

*Este registro constituye la base para justificar la arquitectura final y sirve como respaldo científico del proceso de desarrollo iterativo.*

---

## *5. Implementación de mejoras*

*A pesar de la arquitectura sólida desarrollada, existen varias oportunidades para seguir optimizando el rendimiento del modelo y ampliar sus capacidades. A continuación, se presentan propuestas concretas de mejora, tanto a nivel técnico como metodológico:*

### *I. Mejora en la Representación de Datos:*

- ***Uso de embeddings para variables categóricas:***
  *Actualmente, las variables categóricas se representan como variables binarias (one-hot). Utilizar embeddings aprendibles permitiría capturar relaciones semánticas entre categorías (por ejemplo, entre tipos de experiencia o etiquetas temáticas).*
- ***Procesamiento de texto con NLP:***
  *Incorporar información textual de campos como* ***description, summary o tags*** *mediante modelos de lenguaje (TF-IDF, BERT, etc.) podría enriquecer la representación contextual de cada POI.*

***II. Arquitectura del Modelo:***

- ***Uso de modelos visuales preentrenados (transfer learning):***
  *Reemplazar la red CNN personalizada por una arquitectura preentrenada (como ResNet18, EfficientNet o MobileNet) permitiría aprovechar conocimiento visual aprendido en datasets extensos (ImageNet), acelerando el entrenamiento y mejorando la generalización.*
- ***Atención multimodal:***
  *Incorporar mecanismos de atención (attention layers o transformers multimodales) permitiría ponderar dinámicamente la importancia relativa de la información visual y tabular en función de cada predicción.*

***II. Estrategia de Entrenamiento:***

- ***Técnicas de regularización avanzadas:***
  *Aplicar métodos como Label Smoothing, Early Stopping o Stochastic Weight Averaging puede ayudar a evitar el sobreajuste y mejorar la estabilidad del entrenamiento.*
- ***Ajuste sistemático de hiperparámetros:***
  *Utilizar herramientas como* ***Optuna, Ray Tune o GridSearch*** *para buscar automáticamente combinaciones óptimas de tasa de aprendizaje, arquitectura, tamaño de batch, etc.*
- ***Entrenamiento con class weights:***
  *En caso de que exista desequilibrio entre clases, se podría incorporar pesos personalizados en la función de pérdida para dar mayor importancia a clases minoritarias.*

***VI. Mejora del Pipeline de Evaluación***

- ***Curvas ROC y Matriz de Confusión:***
  *Incluir métricas visuales para entender mejor el comportamiento por clase, especialmente en el caso multiclase.*
- ***Validación cruzada (K-Fold):***
  *Implementar validación cruzada permitiría una evaluación más robusta, especialmente útil si se dispone de un dataset limitado.*

---

## *6. Conclusiones*

### *6.1 Conclusión del modelo Binario*

*Durante la fase experimental de clasificación binaria, se evaluaron múltiples combinaciones de tasas de aprendizaje (**lr**) y épocas (**num_epochs**). Se observó que una tasa intermedia (**0.001**) ofreció la mejor relación entre velocidad de convergencia y capacidad de generalización. Las tasas más bajas proporcionaron mayor estabilidad, aunque sin ganancias significativas al aumentar las épocas, indicando una posible saturación temprana del aprendizaje. La configuración final fue seleccionada en base al mejor desempeño en test (78% de accuracy), consistencia en métricas por clase y ausencia de señales claras de sobreajuste*

I. **Experimento 1: num_epochs=50, lr=0.001** ●
*El **Train Loss** decrece de forma constante.*

- *El **Train Acc** pasa de 50% a casi 78%.*
- *El **Val Acc** sube desde 50% a ~80%, manteniéndose estable en los últimos epochs.*
- ***Test Accuracy: 78%***
- *Buen balance entre precisión y recall.*

***Conclusión**: Este experimento muestra un entrenamiento sano, generalización efectiva y excelente rendimiento en test. La tasa de aprendizaje de 0.001 parece apropiada, ya que logra un rápido descenso del error sin sobreajuste aparente.*

II. **Experimento 2: num_epochs=15, lr=1e-4**

- ***Train Loss** empieza bajo (~0.45) y mejora levemente.*
- ***Train Acc** alcanza 83% pero la Val Acc queda estancada alrededor de ~76–78%.*
- ***Test Accuracy: 75%***
- *Rendimiento equilibrado pero sin mejora notable respecto al anterior.*

***Conclusión:** Con un **learning rate** tan bajo (**1e-4**), el modelo aprende más lentamente. Aunque no hay señales de sobreajuste, no aporta mejora frente a **lr=0.001**, y se estanca más fácilmente. Puede ser útil si se entrena con más épocas.*

III. **Experimento 3:num_epochs=20, lr=1e-4** ●
***Train Loss** baja progresivamente hasta 0.33.*

- ***Train Acc** sube hasta 87%.*
- ***Val Acc** no mejora respecto al experimento anterior (se mantiene entre 79–83%). ● **Test Accuracy: 75%***

- *Ligera mejora de precisión en clase Bajo, ligera caída en Alto.*
- *F1-score general sigue en 0.75.*

***Conclusión:*** *Más épocas permiten refinar el aprendizaje, pero no se observan mejoras claras en generalización. Podría estar ocurriendo un ligero overfitting, ya que Train Acc sube pero Val Acc y Test Acc no lo acompañan.*

### *6.1 Conclusión del modelo Multiclase*

I. ***Experimento 1: num_epochs=15, lr=1e-4*** ●
*Evolución del entrenamiento:*
- *El modelo parte con bajo rendimiento (**Train Acc: 0.27, Val Acc: 0.28)**, pero mejora progresivamente.*
- *Hacia la época 10 en adelante, tanto la precisión en entrenamiento como en validación se estabilizan (~57%).* ● ***Test accuracy: 0.58*** ● *F1 por clase:*
- *Bajo y Muy Bajo tienen menor rendimiento que Alto y Muy Alto.*
- *Esto podría sugerir que el modelo tiene más dificultad en distinguir las clases de menor engagement (quizás por menor representación o mayor ambigüedad visual/contextual).*

***Conclusión:*** *Entrenamiento estable y saludable. El modelo aprende, generaliza moderadamente y no presenta signos de sobreajuste. Buen punto de partida.*

II. ***Experimento 2: num_epochs=20, lr=0.001*** ●
*Evolución:*
- *Mejora más rápida al inicio, llegando al 62% train acc en época 13–14.*
- *Algunas fluctuaciones en **val_loss** (aumenta en épocas 15–17), pero se recupera al final.*

● ***Test accuracy: 0.577*** ●
*F1 por clase:*
- *Muy Alto y Alto mantienen buen rendimiento.*
- *Se observa mejora en la clase Bajo, lo cual sugiere mejor balance que en el experimento anterior.*

***Conclusión:*** *Aprendizaje más rápido con **lr=0.001**. Podría mejorar aún más con más épocas o regularización adicional (dropout, weight decay).*

***III. Experimento 3: num_epochs=50, lr=0.01*** ●

*Evolución:*

- ○ ***Catástrofe desde la primera época: Train Loss y Val Loss*** *se estancan en ~1.386.*
- ○ *Modelo colapsa en una sola clase ("Muy Alto").*

- ***Test accuracy: 0.25***
- *Precision y recall en otras clases: 0.00*
  - ○ *El modelo claramente no está aprendiendo.*

***Conclusión:*** *El learning rate* ***0.01*** *es demasiado alto. El optimizador probablemente está saltando los mínimos, haciendo que el modelo no aprenda nada útil. Este es un caso típico de desbordamiento del gradiente.*

---

***Observaciones sobre el desarrollo****:*

*Este proyecto ha sido desarrollado en un entorno académico y experimental, haciendo uso de buenas prácticas de reutilización y asistencia tecnológica. En particular:*

- *Se* ***reutilizó y adaptó código base proporcionado por el profesorado****, como parte del material de referencia para el desarrollo de proyectos multimodales en PyTorch.*

- *Se hizo uso de herramientas de* ***asistencia por inteligencia artificial*** *para mejorar la redacción, depurar errores y optimizar fragmentos de código.*

*Estas herramientas fueron utilizadas como apoyo para el aprendizaje, verificación de conceptos y mejora de la productividad, manteniendo siempre una comprensión crítica del proceso.*